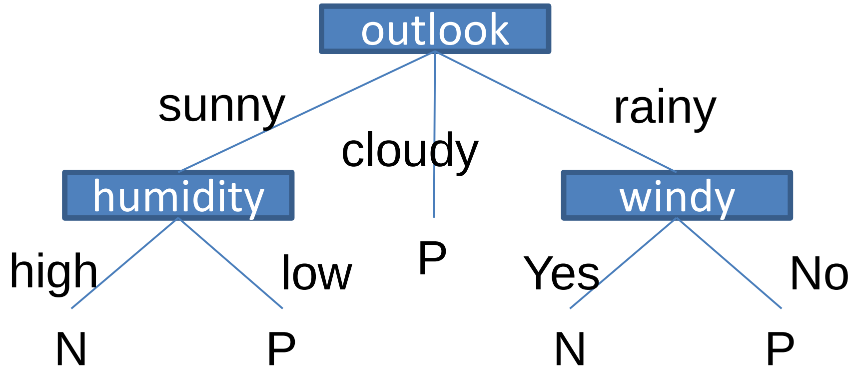
Decision Tree/Random Forest

Prof. Chia-Yu Lin
Yuan Ze University

2021 Spring

A Decision-Tree Based Classification

- A decision tree of whether going to play tennis or not:

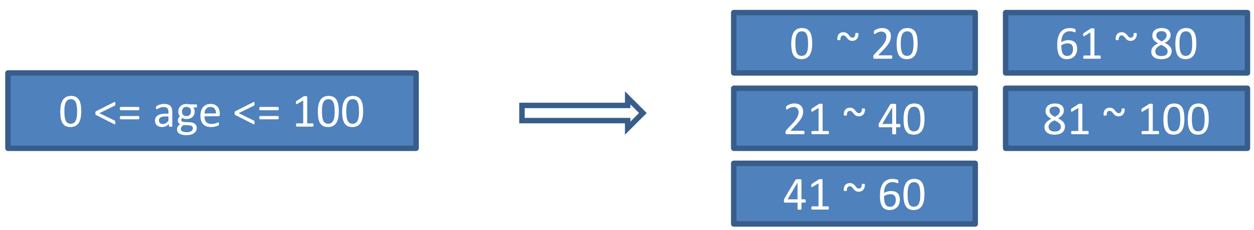


- Tree can explain the rule for classification.
- **ID-3** and its extended version **C4.5** (Quinlan'93): A **top-down** decision tree generation algorithm

Algorithm for Decision Tree Induction (1/2)

- Basic algorithm (a **greedy algorithm**)
  - Tree is constructed in a top-down recursive divide-and-conquer manner.
  - Attributes are categorical.

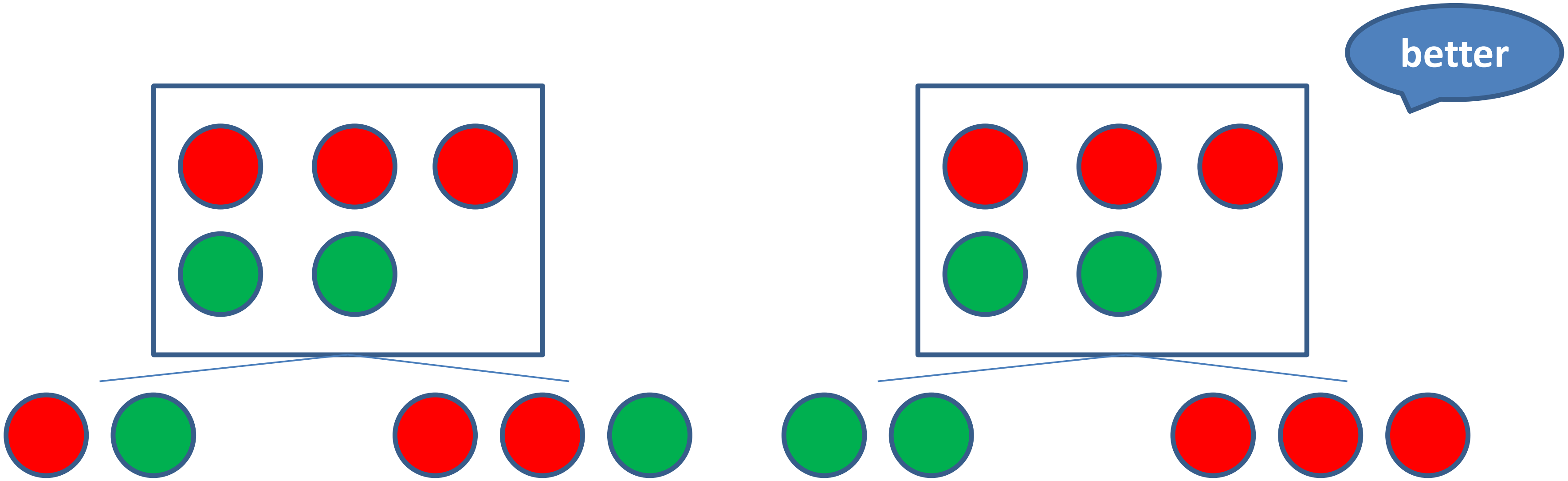
    (if an attribute is a continuous number, it needs to be discretized in advance.) E.g.



  - At start, all the training examples are at the root.
  - Examples are partitioned recursively based on selected attributes.

Algorithm for Decision Tree Induction (2/2)

- Basic algorithm (a **greedy algorithm**)
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain): maximizing an information gain measure,
  - i.e., **favoring the partitioning which makes the majority of examples belong to a single class**.

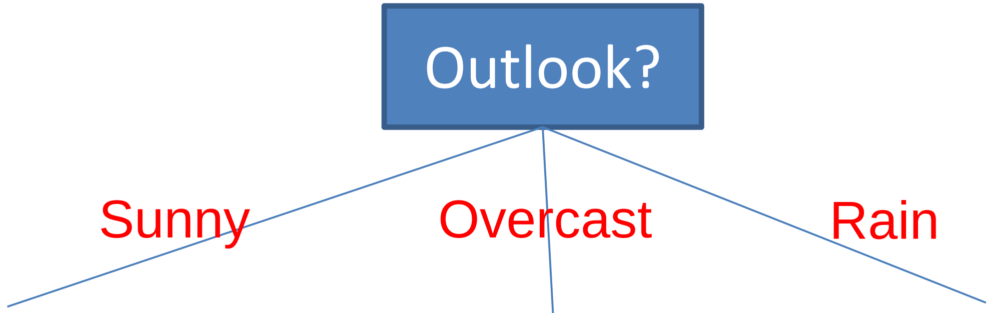


Algorithm for Decision Tree Induction (2/2)

- Basic algorithm (a **greedy algorithm**)
  - Conditions for stopping partitioning:
    - All samples for a given node belong to the same class
    - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
    - There are no samples left

Decision Tree Induction: Training Dataset

Day	Outlook	Temp.	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



Day	Outlook	Temp.	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes

Day	Outlook	Temp.	Humidity	Wind	Play Tennis
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D10	Rain	Mild	Normal	Strong	Yes
D14	Rain	Mild	High	Strong	No

Day	Outlook	Temp.	Humidity	Wind	Play Tennis
D3	Overcast	Hot	High	Weak	Yes
D7	Overcast	Cool	Normal	Weak	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes

Primary Issues in Tree Construction (1/2)

- **Split criterion**: *Goodness function*
  - Used to select the attribute to be split at a tree node during the tree generation phase
  - Different algorithms may use different goodness functions:
    - Information gain (used in ID3/C4.5)
    - Gini index (used in CART)

Primary Issues in Tree Construction (2/2)

- **Branching scheme**:
  - Determining the tree branch to which a sample belongs
  - Binary vs. *k-ary* splitting

- When to stop the further splitting of a node? e.g. impurity measure

- Labeling rule: a node is labeled as the class to which most samples at the node belongs.

How to Use a Tree?

- Directly
  - Test the attribute value of unknown sample against the tree.
  - A path is traced from root to a leaf which holds the label.
- Indirectly
  - Decision tree is converted to classification rules.
  - One rule is created for each path from the root to a leaf.
  - **IF-THEN** is easier for humans to understand .

Attribute Selection Measure: Information Gain (ID3/C4.5)

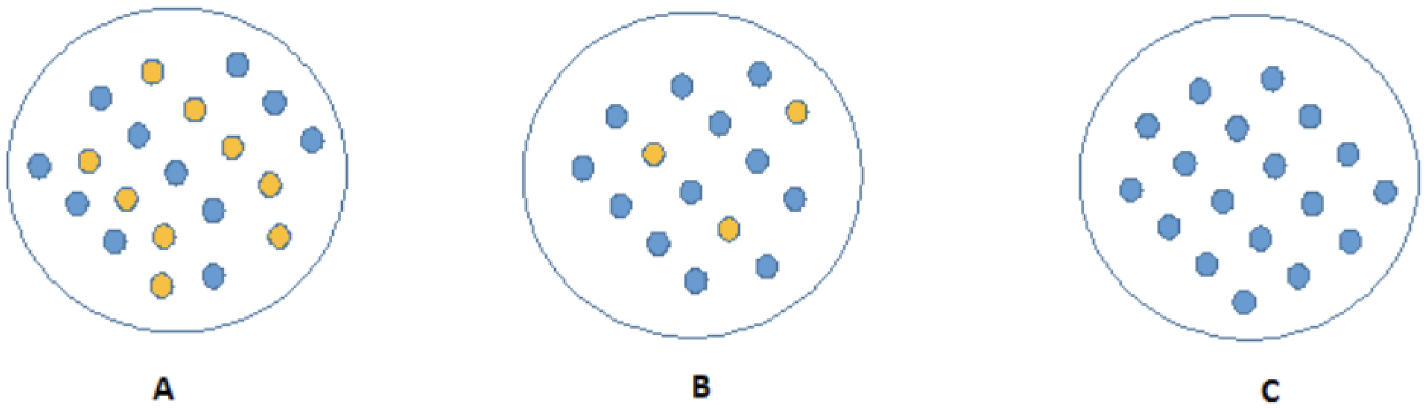
- **Select the attribute with the highest information gain**
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i,D}|/|D|$
- Expected information (**entropy**) needed to classify a tuple in D:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- Expected information (**entropy**):
  - Entropy is a measure of how "mixed up" an attribute is.
  - It is sometimes equated to the purity or impurity of a variable.
  - High Entropy means that we are sampling from a uniform (boring) distribution.

Example

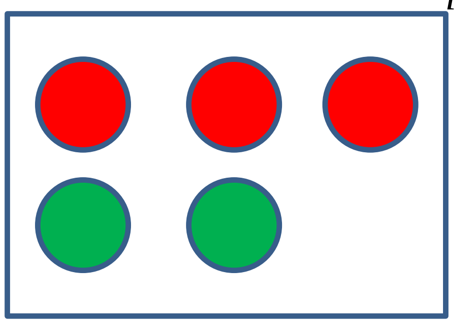
- Which one can be explained with the least information?

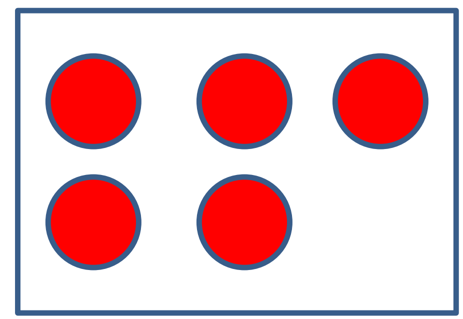


Expected Information (Entropy)

- Expected information (**entropy**) needed to classify a tuple in D:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i) \quad \text{(m: number of labels)}$$



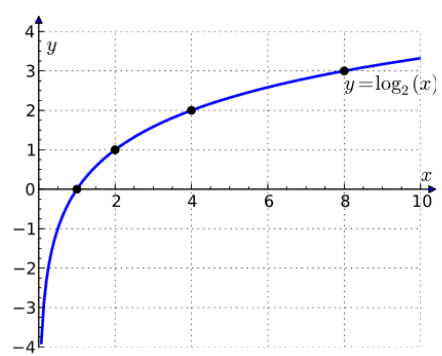


$$Info(D) = I(3,2) = -\frac{3}{5}\log_2(\frac{3}{5}) - \frac{2}{5}\log_2(\frac{2}{5})$$

$$\approx -\frac{3}{5}\times(-0.737) - \frac{2}{5}\times(-1.322)$$

$$Info(D) = I(5,0) = -\frac{5}{5}\log_2(\frac{5}{5}) - \frac{0}{5}\log_2(\frac{0}{5})$$

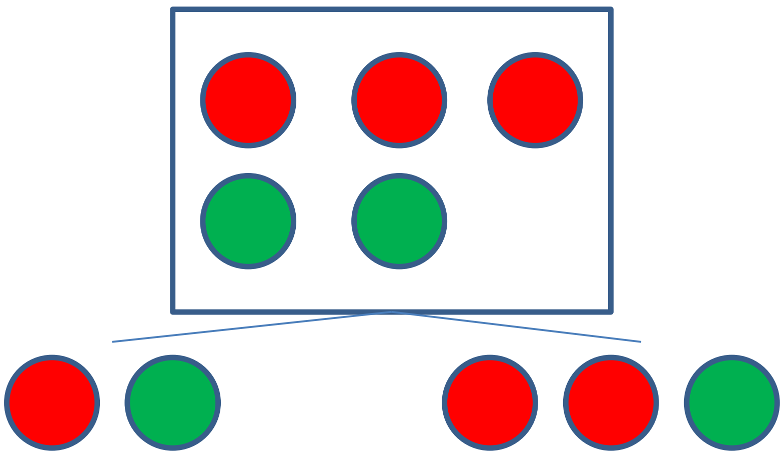
$$= 0 - 0 = 0$$



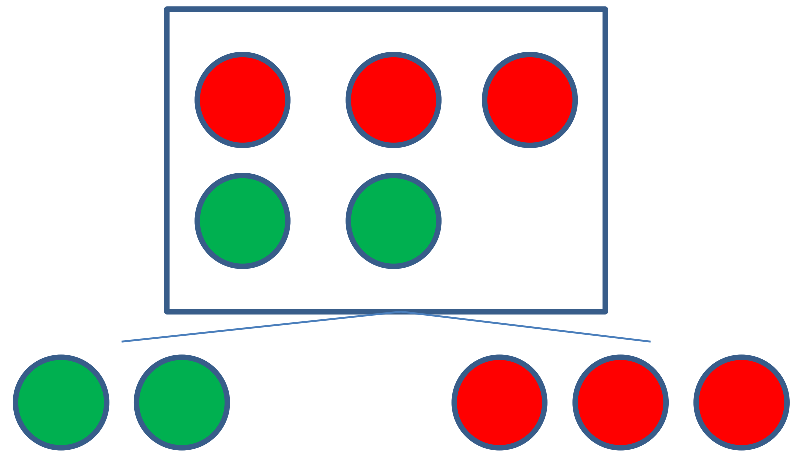
Expected Information (Entropy)

- Information needed (after using A to split D into v partitions) to classify D:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times I(D_j)$$



$$Info(D) = \frac{2}{5} Info(1,1) + \frac{3}{5} Info(2,1)$$



$$Info(D) = \frac{2}{5} Info(2,0) + \frac{3}{5} Info(3,0)$$

Attribute Selection Measure: Information Gain (ID3/C4.5)

- **Select the attribute with the highest information gain**
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i,D}|/|D|$
- Expected information (**entropy**) needed to classify a tuple in D:
$$Info(D) = I(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$
- Information needed (after using A to split D into v partitions) to classify D:
$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times I(D_j)$$
- Information gained by branching on attribute A
$$Gain(A) = Info(D) - Info_A(D)$$

Example: Information Gain (1/2)

- Class P: Play Tennis = “yes”
- Class N:Play Tennis= “no”

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$$

$$Info_{Outlook}(D) = \frac{5}{14}I(2,3) + \frac{4}{14}I(4,0) + \frac{5}{14}I(3,2) = 0.694$$

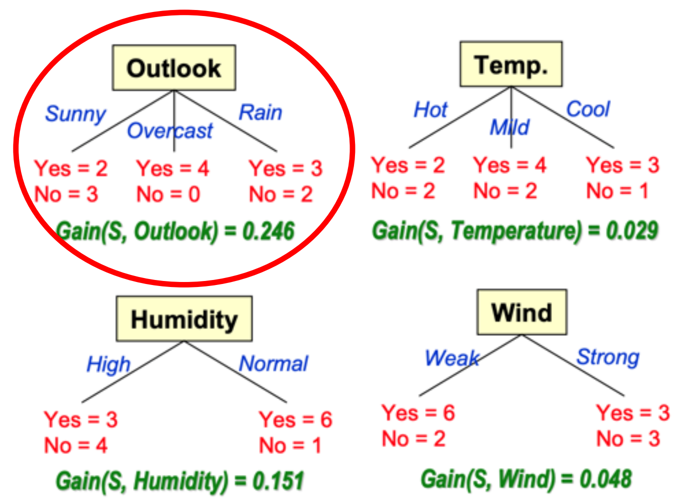
$\frac{5}{14}I(2,3)$ means “Sunny” has 5 out of 14 samples, with 2 yes’es and 3 no’s.

Hence

$$Gain(Outlook) = Info(D) - Info_{Outlook}(D) = 0.246$$

Day	Outlook	Temp.	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

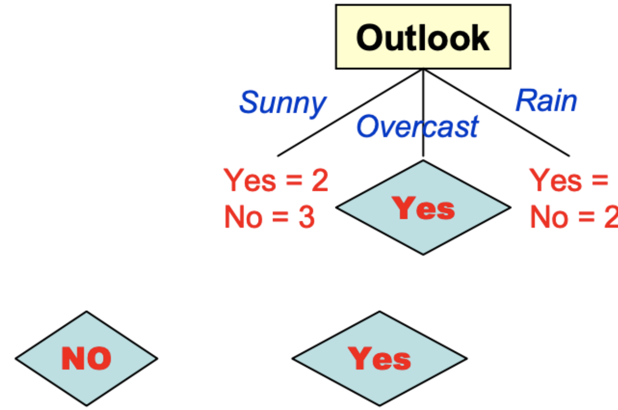
Similarly,

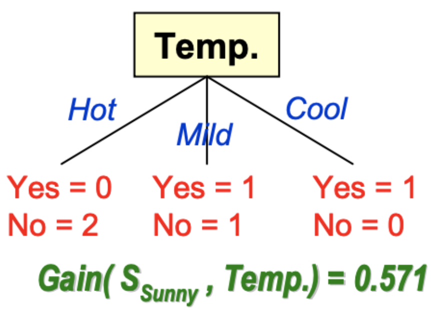


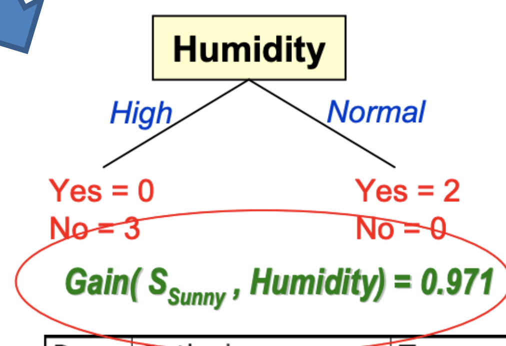
Example: Information Gain (2/2)

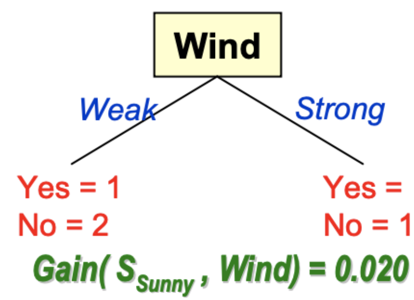
List the data which "Outlook=Sunny" and compute the information gain.

Day	Outlook	Temp.	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes





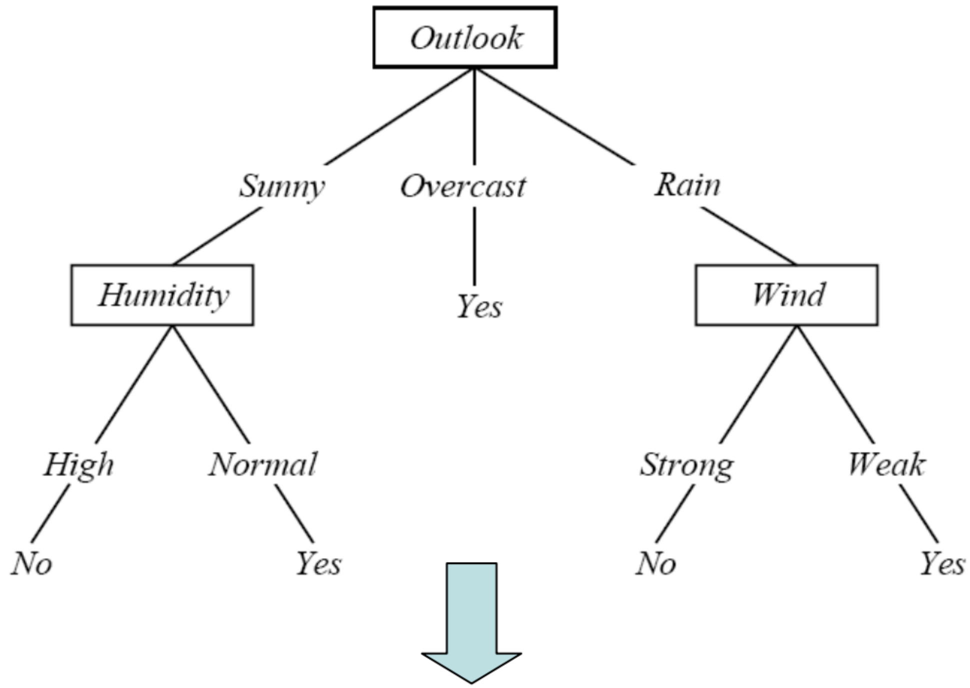




List the data which "Outlook=Rain" and compute the information gain.

Day	Outlook	Temp.	Humidity	Wind	Play Tennis
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D10	Rain	Mild	Normal	Strong	Yes
D14	Rain	Mild	High	Strong	No

Decision Tree



Rule:

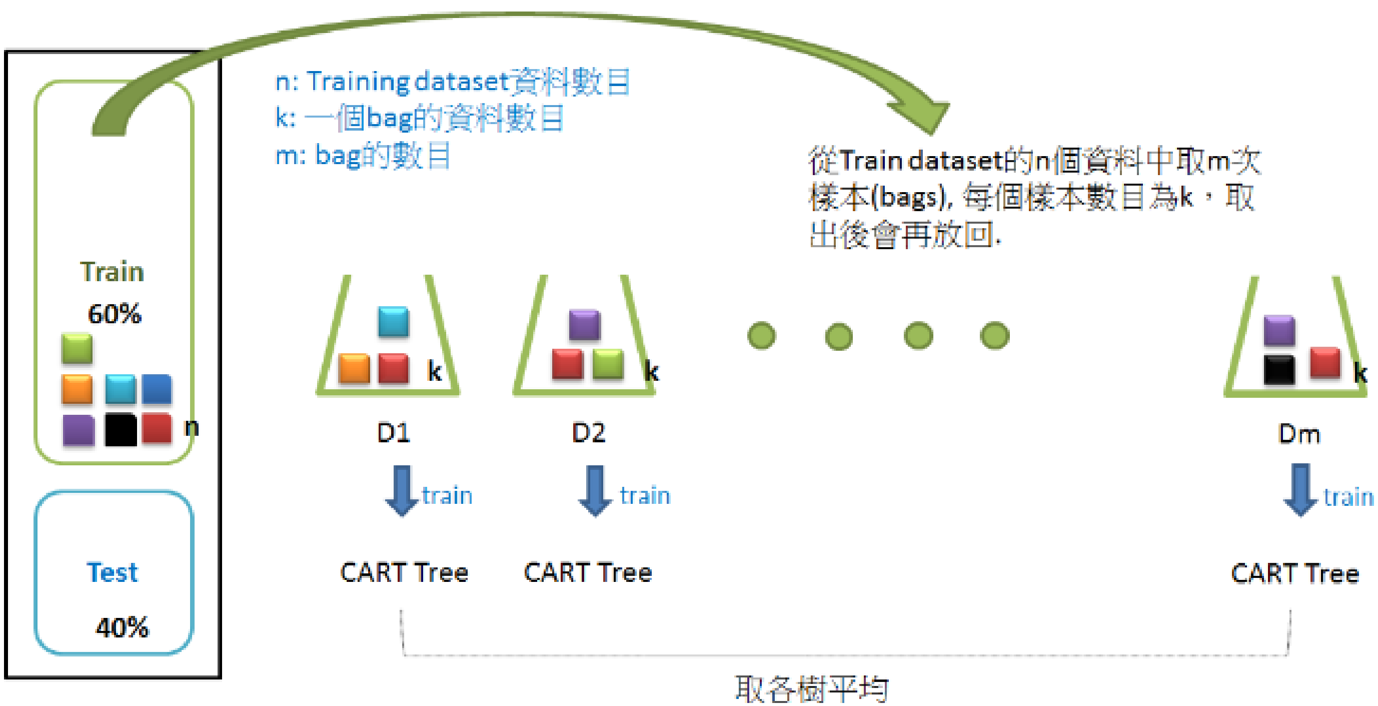
If Outlook = Sunny and Humidity = High Then Play Tennis = No
If Outlook = Sunny and Humidity = Normal Then Play Tennis = Yes
If Outlook = Overcast Then Play Tennis = Yes
If Outlook = Rain and Wind = Strong Then Play Tennis = No
If Outlook = Rain and Wind = Weak Then Play Tennis = Yes

Random Forest (1/2)

- Random Forest:
  - Each classifier in the ensemble is a ***decision tree*** classifier and is generated using a random selection of attributes at each node to determine the split
  - During classification, each tree votes and the most popular class is returned
  - More accurate.
  - Ensemble Method
    - There must be difference between every classifier.
    - The accuracy of each classifier has be greater than 0.5.
  - Bagging
  - Boosting

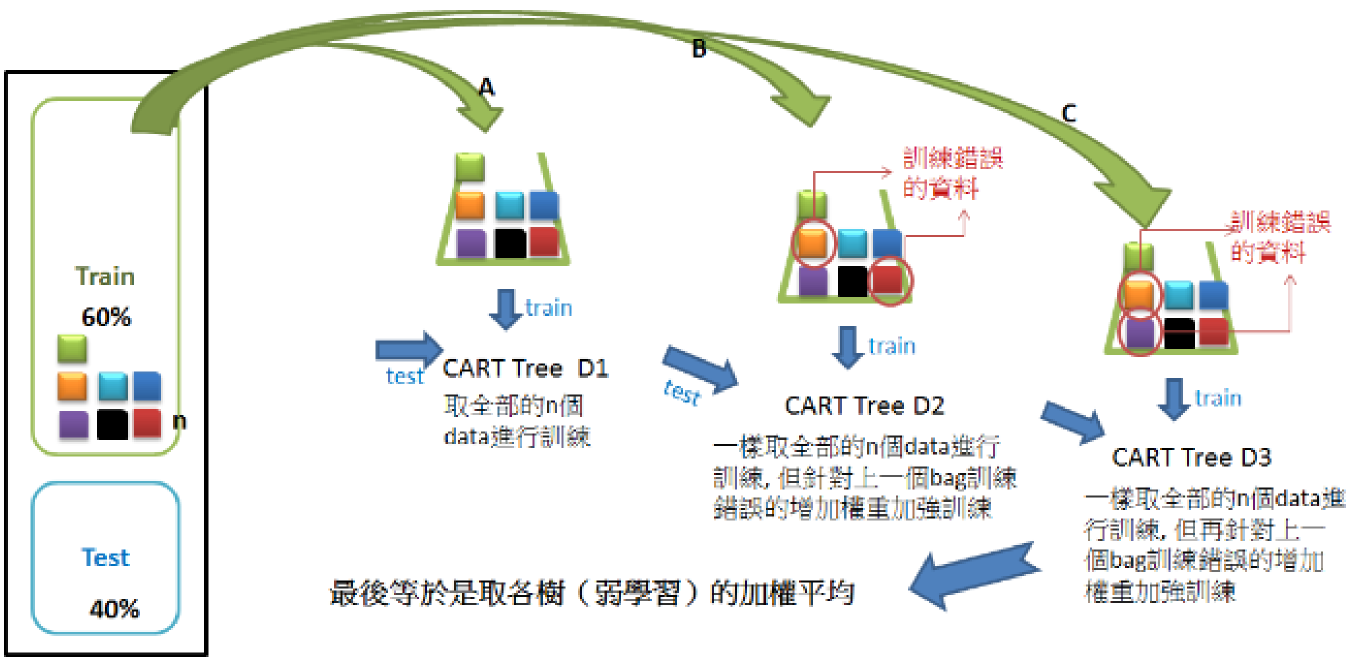
Bagging

- **B**ootstrap **agg**regat**ing**
  - Takes K samples from training set and construct K classifier.
  - Put K samples back to the training set.
  - And do it again.



Boosting

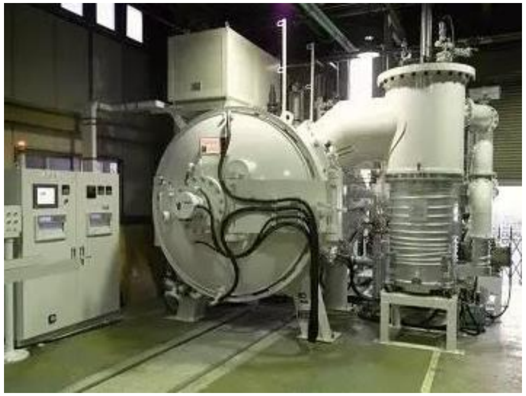
- Similar to Bagging method.
- Emphasis on the error part to improve the overall efficiency.
- The key point is to gradually train a large number of weak learning classifiers (the efficiency is not so good) into a stronger classifier.



Lab: Random Forest

Dataset

- **以熱壓爐溫度預測成化曲線的分類**
  - 2019全國智慧製造大數據分析競賽
  - 數據為熱壓爐成化加工過程所量測的溫度數據
    - 成化：複合材料加工至硬化的溫度
    - 依照機台型號可以分為 8 類
    - 目標：訓練multi-class分類模型，可以準確分 8 類



SVM Result

- Result from SVM
  - Accuracy: 99.02%

```
Macro-average: 0.990224871165917
Micro-average: 0.9902354968408961
              precision    recall  f1-score   support

           0       1.00      0.97      0.99       145
           1       1.00      0.99      1.00       207
           2       1.00      0.97      0.99       119
           3       1.00      1.00      1.00       238
           4       0.99      0.99      0.99       256
           5       1.00      1.00      1.00       264
           6       1.00      0.99      0.99       240
           7       0.95      0.99      0.97       272

   micro avg       0.99      0.99      0.99      1741
   macro avg       0.99      0.99      0.99      1741
weighted avg       0.99      0.99      0.99      1741

[[141   0   0   0   0   0   0   4]
 [  0 205   0   0   0   0   0   2]
 [  0   0 116   0   0   0   0   3]
 [  0   0   0 238   0   0   0   0]
 [  0   0   0   0 253   0   0   3]
 [  0   0   0   0   0 264   0   0]
 [  0   0   0   0   0   0 237   3]
 [  0   0   0   0   2   0   0 270]]
Accuracy: 99.02%
Average = macro
precision: 0.9924406604747162
recall: 0.9882462727680715
recall: 0.9882462727680715
F1-score: 0.9931487344522549


Average = micro
precision: 0.9902354968408961
recall: 0.9902354968408961
F1-score: 0.9938800489596082


Average = weighted
precision: 0.9906239904589667
recall: 0.9902354968408961
F1-score: 0.9938672003987704
```

Use Different Classifier

- Random Forest

```
class sklearn.ensemble.RandomForestClassifier(n_estimators=10, crite-rion='gini', max_depth=None,
min_samples_split=2, min_samples_leaf=1,
min_weight_fraction_leaf=0.0,
max_features='auto',
max_leaf_nodes=None, bootstrap=True,
oob_score=False, n_jobs=1, ran-dom_state=None, verbose=0,
warm_start=False, class_weight=None)
```

- n_estimators：決策樹的個數。
- criterion: ” gini” or “entropy” 來選擇合適的節點，預設是gini 。
- max_depth: 樹的最大深度，預設為None，這樣建樹的時候會一個葉節點屬於一個類別。
- min_samples_split:根據屬性畫分節點時，每個畫分最少的樣本數。
  min_samples_leaf:葉子節點最少的樣本數。
- max_features: 屬性的最大個數。
- max_leaf_nodes:葉子數的最大樣本數，預設為None。
- bootstrap：是否有放回的采样。 (True:有放回的隨機採樣)
- oob_score：希望用袋外數據測試時設定為True。
- n_jobs=1：並行job個數。 1=不並行；n：n個並行；-1：CPU有多少core，就啟動多少job
- warm_start=False：熱啟動，決定是否使用上次分類的結果，然後增加新的。
- class_weight=None：各個label的權重。

Coding

```
#random forest
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC

#10-fold cross-validation
kfold = KFold(10, True)
predicted = []
expected = []

for train, test in kfold.split(dataset):
    X_train= dataset.iloc[train]
    Y_train = label.iloc[train]
    X_test = dataset.iloc[test]
    Y_test = label.iloc[test]
    forest = ensemble.RandomForestClassifier(n_estimators = 100)
    forest.fit(X_train,Y_train)
    expected.extend(Y_test)
    predicted.extend(forest.predict(X_test))
```

Random Forest Result

```
Macro-average: 0.9955076920244802
Micro-average: 0.9954049396898335
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       145
           1       1.00      0.99      0.99       207
           2       1.00      1.00      1.00       119
           3       0.99      1.00      1.00       238
           4       0.99      0.99      0.99       256
           5       1.00      1.00      1.00       264
           6       1.00      1.00      1.00       240
           7       1.00      1.00      1.00       272

    accuracy                           1.00      1741
   macro avg       1.00      1.00      1.00      1741
weighted avg       1.00      1.00      1.00      1741

[[143   0   0   0   2   0   0   0]
 [  0 205   0   2   0   0   0   0]
 [  0   0 119   0   0   0   0   0]
 [  0   0   0 238   0   0   0   0]
 [  0   1   0   0 254   0   0   1]
 [  0   0   0   0   0 264   0   0]
 [  0   0   0   0   0   1 239   0]
 [  0   0   0   0   0   0   1 271]]
Accuracy: 99.54%
Average = macro
precision: 0.9959228844467581
recall: 0.9951111779435038
F1-score: 0.9951848692658576


Average = micro
precision: 0.9954049396898335
recall: 0.9954049396898335
F1-score: 0.9945155393053017
```